https://github.com/gitgik/distributed-system-design/blob/master/designing-cloud-storage.ipynb

Volume: Organizations collect data from a variety of sources, including business transactions, smart (IoT) devices, industrial equipment, videos, social media and more. In the past, storing it would have been a problem – but cheaper storage on platforms like data lakes and Hadoop have eased the burden.
https://www.linkedin.com/pulse/all-big-data-md-sahil/

An inherent limitation of distributed storage systems is defined by the CAP theorem. The theorem states that a distributed system cannot maintain Consistency, Availability and Partition Tolerance (the ability to recover from a failure of a partition containing part of the data). It has to give up at least one of these three properties. Many distributed storage systems give up consistency while guaranteeing availability and partition tolerance.
https://github.com/PotatoSpudowski/The-centralized-guide-to-distributed-storage-and-processing-of-big-data

Code:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
```

2. Set up the Hadoop configuration:

```
Configuration conf = new Configuration();
conf.set("fs.defaultFS", "hdfs://localhost:9000");
```

3. Create a file system object:

```
FileSystem fs = FileSystem.get(conf);
```

4. Create a path object for the file you want to store:

```
Path inputPath = new Path("/path/to/input/file.txt");
```

5. Create a path object for the location where you want to store the file:

```
Path outputPath = new Path("/path/to/output/");
```

6. Check if the output path already exists:

```
if (fs.exists(outputPath)) {
    System.out.println("Output path already exists.");
    return;
}
```

7. Create the output directory:

```
fs.mkdirs(outputPath);
```

8. Copy the input file to the output directory:

```
fs.copyFromLocalFile(inputPath, outputPath);
```

9. Close the file system object:

```
fs.close();
```

Storing huge data using big data concepts involves using distributed systems that are designed to handle large amounts of data by splitting the data across multiple machines. Here are some common methods for storing huge data using big data concepts:

1. Hadoop Distributed File System (HDFS): HDFS is a distributed file system that is used to store and manage large data sets across multiple machines. HDFS provides a fault-tolerant architecture that ensures data is not lost in the event of a hardware failure.
2. NoSQL Databases: NoSQL databases, such as Apache Cassandra or MongoDB, are designed to handle large volumes of unstructured or semi-structured data. These databases use a distributed architecture that allows them to scale horizontally across multiple machines.

```
// Import the necessary Cassandra libraries

import com.datastax.driver.core.Cluster;
```

```java
import com.datastax.driver.core.Session;


// Set up the Cassandra cluster

String node = "localhost";

int port = 9042;

Cluster cluster =
Cluster.builder().addContactPoint(node).withPort(port).build();


// Connect to the Cassandra cluster

Session session = cluster.connect();


// Create the keyspace

String keyspace = "my_keyspace";

String replicationStrategy = "SimpleStrategy";

int replicationFactor = 1;

session.execute("CREATE KEYSPACE IF NOT EXISTS " + keyspace +

        " WITH replication = {'class':'" + replicationStrategy + "',
'replication_factor':" + replicationFactor + "};");


// Create the table

String table = "my_table";

session.execute("CREATE TABLE IF NOT EXISTS " + keyspace + "." + table + "
(id UUID PRIMARY KEY, name text, age int);");
```

```
// Insert data into the table

String id = UUID.randomUUID().toString();

String name = "John";

int age = 30;

session.execute("INSERT INTO " + keyspace + "." + table + " (id, name, age)
VALUES (" + id + ", '" + name + "', " + age + ");");



// Query the data from the table

ResultSet results = session.execute("SELECT * FROM " + keyspace + "." + table
+ ";");

for (Row row : results) {

    System.out.println("ID: " + row.getUUID("id") + ", Name: " +
row.getString("name") + ", Age: " + row.getInt("age"));

}



// Close the session and cluster

session.close();

cluster.close();
```

3. Cloud Storage: Cloud storage services, such as Amazon S3 or Google Cloud Storage, are designed to store and manage large data sets in a distributed environment. These services provide a pay-as-you-go model that allows you to scale your storage needs up or down based on demand.

```
// Import the necessary AWS SDK libraries

import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.auth.BasicAWSCredentials;

import com.amazonaws.client.builder.AwsClientBuilder;

import com.amazonaws.services.s3.AmazonS3;

import com.amazonaws.services
```

4. Distributed File Systems: Distributed file systems, such as GlusterFS or Ceph, provide a way to store and manage large data sets across multiple machines. These systems are designed to handle both structured and unstructured data and provide high availability and fault tolerance.

To store huge data using big data concepts, it is important to choose a storage solution that is designed to handle the specific data requirements of your organization. Additionally, it is important to ensure that your storage solution can scale horizontally to handle the increasing data volumes that your organization will likely generate over time

https://www.simplilearn.com/tutorials/hadoop-tutorial/what-is-hadoop

https://cloudxlab.com/blog/introduction-to-big-data-and-distributed-computing/
https://cloudxlab.com/blog/big-data-solution-apache-hadoop-and-spark/
https://www.codingninjas.com/codestudio/library/why-do-we-need-distributed-computing-for-big-data
https://medium.com/bazaar-tech/apache-spark-data-cleaning-using-pyspark-for-beginners-eeeced351ebf
https://medium.com/bazaar-tech/apache-spark-data-cleaning-using-pyspark-for-beginners-eeeced351ebf